Ensemble Learning and Classification

Ensemble Classifiers

Basic Idea

Instead of selecting a “best” hypothesis, keep many and combine their outputs

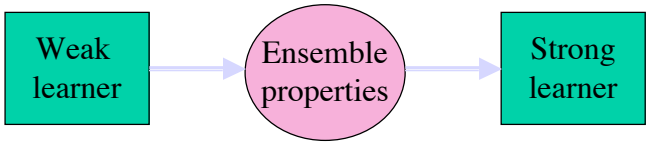
Definitions

- Strong learner

$$\Pr\{\|h - \omega\| \leq \varepsilon\} \geq 1 - \delta$$

- Weak learner

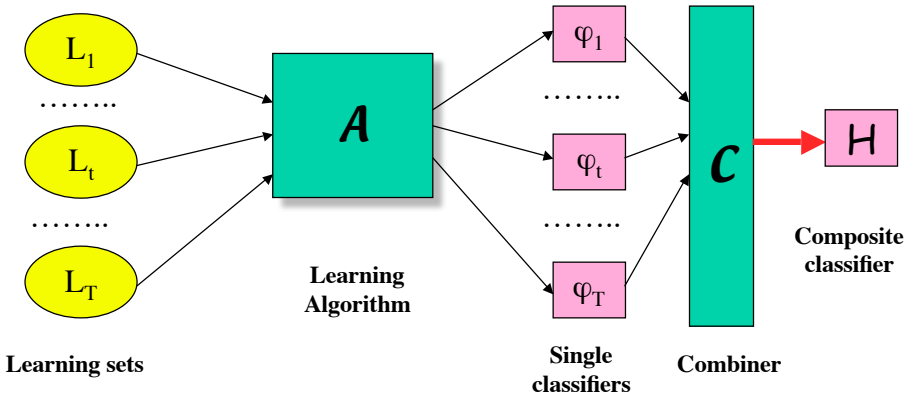
$$\varepsilon \leq 1/2 - \gamma \qquad (\gamma \to 0^{+})$$



$$\| h - \omega \| = \frac{1}{N} \sum_{x \in X} I_{h(x) \neq \omega(x)}$$

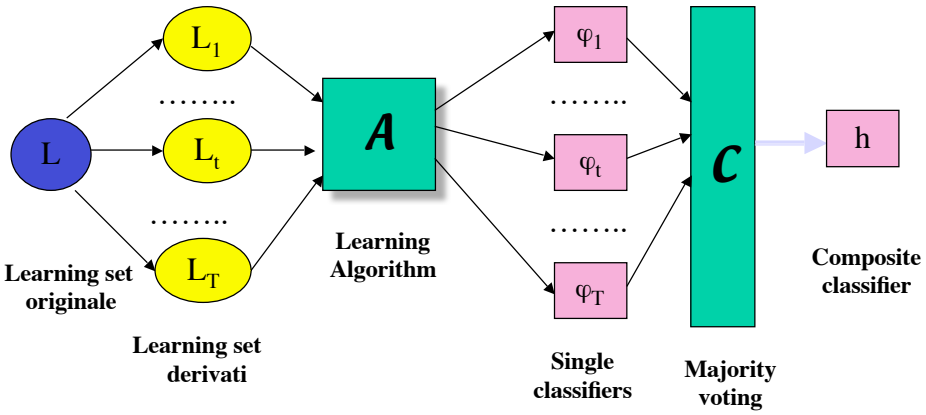
Se $h, \omega \in \{0,1\}$: $\|h-\omega\| = \frac{1}{N} \sum_{x \in X} | h(x) - \omega(x) |$

Ensemble Learning and Classification

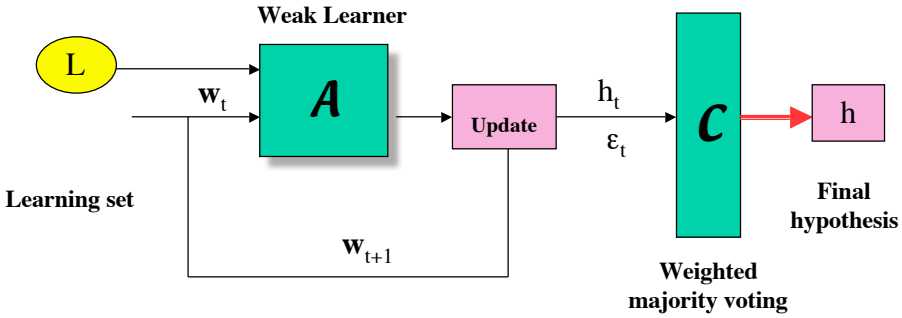


Methods differ with respect to the strategy for generating the learning sets L_t, to the nature of A, and to the combination rule

Bagging (Bootstrap Aggregation)



Boosting



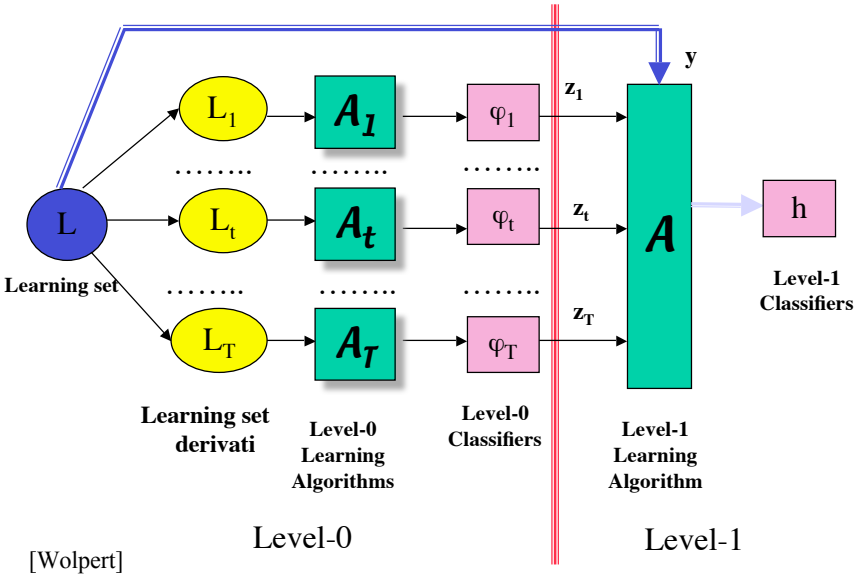
$L = \{(x_i, y_i) \mid 1 \le i \le N\}$

Distribuzione di probabilità P su $X \times Y$: Non nota e data dal mondo

Distribuzione di probabilità D su L : Controllata dall'utente

Pesatura variabile degli esempi di apprendimento
Si pesano di più gli esempi misclassificati

Stacked Generalization



[Wolpert]

Analisi della Stacked Generalization

I learning set derivati L_t sono ottenuti per partizione all'incirca uniforme di L in T sottoinsiemi disgiunti

La stacked generalization può essere fatta in parallelo

Gli algoritmi di livello 0 applicati agli L_t possono essere diversi tra loro

Dato un elemento $(\mathbf{x_i}, y_i) \in L$, ogni classificatore φ_t di livello 0 ne fornisce una classificazione $z_{t,i} = \varphi_t(\mathbf{x_i})$.

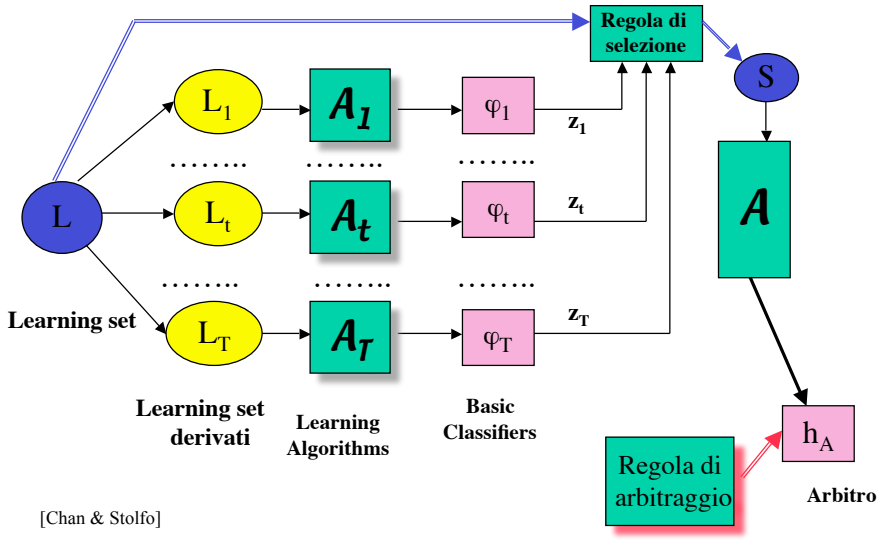
Per ogni $\mathbf{x_i}$, si forma un vettore di nuovi attributi

$$\mathbf{x_i} \Rightarrow (z_{1i}, z_{2i}, \ldots, z_{Ti}, y_i),$$

che è il nuovo dato in ingresso all'apprendimento di livello 1.

Per classificare, si usano in sequenza tutti i classificatori φ_t
e poi h(x) sui loro risultati

Meta-Learning (Arbitraggio)

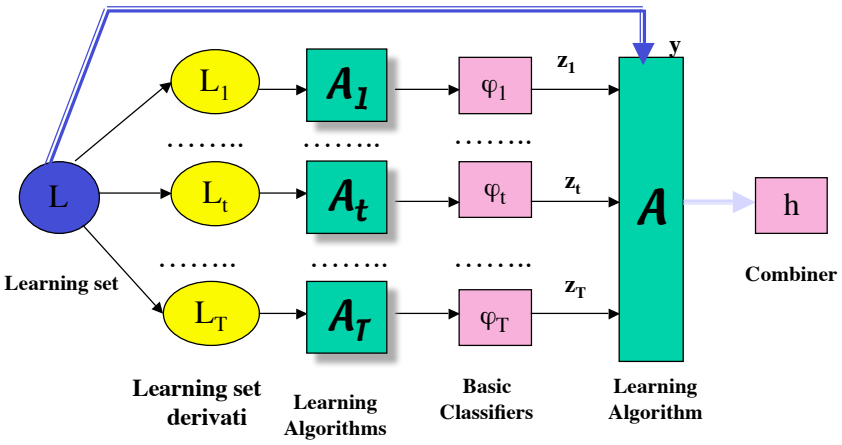


[Chan & Stolfo]

Analisi del Meta-Learning (Arbitraggio)

- Il learning set è partizionato
- L'arbitro sceglie tra classificazioni contraddittorie mediante l'uso di una regola di arbitraggio
  - **"Scegliere la classe che ha ottenuto la maggioranza dei voti.**
  - **In caso di parità, accettare la classficazione dell'arbitro"**
- L'arbitro è appreso da un training set S che è un sottoinsieme di L. Il set S è scelto secondo una regola di selezione
  - **"Mettere in S quei dati originari di L per cui i classificatori singoli non sono riusciti a esprimere una maggioranza per una classe"**

Meta-Learning (Combinazione)

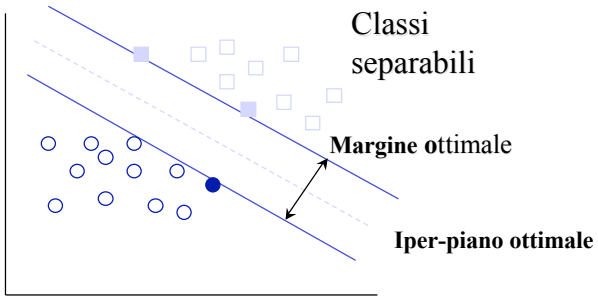


Analisi del Meta-Learning (Combinazione)

- Class-Combiner
  - I dati per imparare h(x) sono vettori che contengono le T classificazioni fornite dai classificatori singoli, più la classe corretta (come per la stacked generalization)
- Class-Attribute-Combiner
  - I dati per imparare h(x) sono vettori che contengono le T classificazioni fornite dai classificatori singoli e la classe corretta, in aggiunta al vettore orignario degli attributi

Margine

Classificazione mediante un iper-piano in uno spazio multi-dimensionale diverso a quello originale. L'iperpiano separa in modo ottimale due classi



$$E(P_{err}) = \frac{\text{E(Numero di vettori di supporto)}}{\text{Numero di esempi di apprendimento}}$$

Bias-variance decomposition

- Theoretical tool for analyzing how much *specific* training set affects performance of a classifier
  - Total expected error: bias + variance
  - The *bias* of a classifier is the expected error of the classifier due to the fact that the classifier is not perfect
  - The *variance* of a classifier is the expected error due to the particular training set used

Bagging

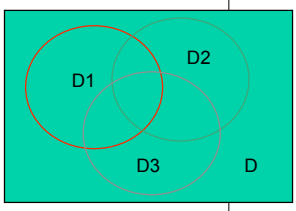
Bootstrap Estimation

- Repeatedly draw *n* samples from *D*
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance

Bagging classifiers

Model generation

```
Let n be the size of the training set.
For each of t iterations:
  Sample n instances with replacement from the
   training set.
  Apply the learning algorithm to the sample.
  Store the resulting model.
```



Classification

```
For each of the t models:
  Predict class of instance using model.
Return class that was predicted most often.
```

Proprietà del Bagging {J classi}

Il bagging può essere fatto in parallelo

$$n_j = |\{h_t(x) \mid h_t(x) = j\}| \qquad 1 \le j \le J$$

$$h(x) = \text{Arg} \max_{1 \le j \le J} n_j$$

Negli esperimenti Breiman ha trovato $T \approx 10$

Why does bagging work?

- Bagging reduces variance by voting/ averaging, thus reducing the overall expected error
  - In the case of classification there are pathological situations where the overall error might increase
  - Usually, the more classifiers the better

Boosting

Adaboost - Adaptive Boosting

- Instead of sampling, re-weight
  - Previous weak learner has only 50% accuracy over new distribution
- Can be used to learn weak classifiers
- Final classification based on weighted vote of weak classifiers

Adaboost – How does it work?

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Adaboost – How does it work?

Base Learner Job:

- Find a base Hypothesis:

$$h_t : X \to \mathbb{R}$$

- Minimize the error:

$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i).$$

- Choose a_t

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Choosing Alpha

- Choose alpha to minimize Z

$$Z(\alpha) = Z = \sum_i D(i) e^{-\alpha u_i}$$

- Results in 50% error rate in latest weak learner

$$\mathrm{E}_{i \sim D_{t+1}} [y_i h_t(x_i)] = 0$$

- In general, compute numerically

The algorithm core

- The main objective is to minimize $\varepsilon_{tr} = \frac{1}{m}|\{i : H(x_i) \neq y_i\}|$
- It can be upper bounded by $\varepsilon_{tr}(H) \leq \prod_{t=1}^{T} Z_t$

How to set α_t?

- Select α_t to greedily minimize $Z_t(\alpha)$ in each step
- $Z_t(\alpha)$ is convex differentiable function with one extremum

  $\Rightarrow$ $h_t(x) \in \{-1, 1\}$ then optimal $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
  where $r_t = \sum_{i=1}^{m} D_t(i)h_t(x_i)y_i$
- $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq 1$ for optimal α_t

  $\Rightarrow$ Justification of selection of h_t according to ϵ_t

Reweighting

Effect on the training set

Reweighting formula:

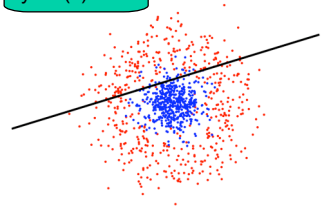
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q}$$

$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

y * h(x) = 1

y * h(x) = -1

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples



Reweighting

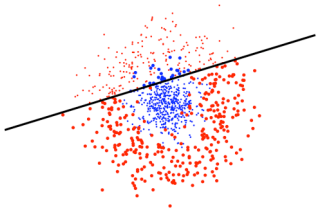
Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q}$$

$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

⇒ Increase (decrease) weight of wrongly (correctly) classified examples



In this way, AdaBoost "focused on" the informative or "difficult" examples.

Reweighting

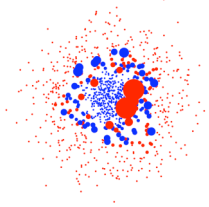
Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q}$$

$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

⇒ Increase (decrease) weight of wrongly (correctly) classified examples



In this way, AdaBoost "focused on" the informative or "difficult" examples.

Remarks on Boosting

- Boosting can be applied without weights using re-sampling with probability determined by weights;
- Boosting decreases exponentially the training error in the number of iterations;
- Boosting works well if base classifiers are not too complex and their error doesn't become too large too quickly!

Proprietà dell'Adaboost

Siano r le h_t con valore 1 e sia $\varepsilon_t \approx \varepsilon$:

$$r \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) \geq \frac{1}{2} T \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) \quad ==> \quad r \geq T/2$$

Se $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_T$ sono gli errori delle ipotesi parziali, si ha:

$$\varepsilon = \Pr\{h(x_i) \neq \omega(x_i)\} \leq \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)}$$

Se $e_t < 1/2 - g$ per ogni t, avremo:

$$e < e^{-2 T \gamma^2}$$

Per raggiungere questo errore occorre che sia almeno:

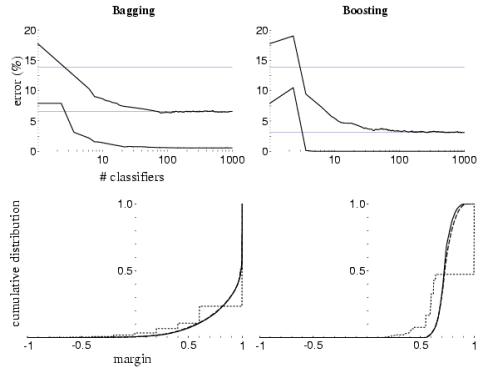
$$T \approx \left\lceil \frac{1}{2\gamma^2} \ln \frac{1}{\varepsilon} \right\rceil$$

Bound on Training Error (Schapire)

$$\begin{aligned}\frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] &\leq \frac{1}{m}\sum_i \exp(-y_i f(x_i)) \\ &= \sum_i \left(\prod_t Z_t\right) D_{T+1}(i) \\ &= \prod_t Z_t \,.\end{aligned}$$

$$Z_t = \sum_i D_t(i)\exp(-\alpha_t y_i h_t(x_i))$$

Adaboost as Margin Maximization (Schapire)



Maximizing the margin…

- But Adaboost doesn't necessarily maximize the margin on the test set (Ratsch)
- Ratsch proposes an algorithm (Adaboost*) that does so

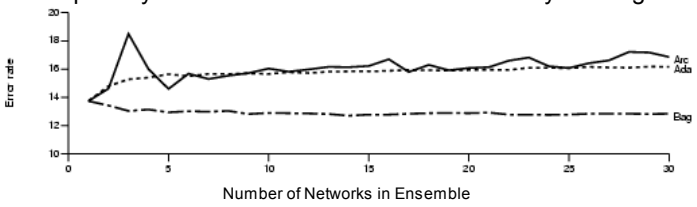
	C4.5	AdaBoost	Marginal AdaBoost	AdaBoost*
E_{gen}	$7.4 \pm 0.11\%$	$4.0 \pm 0.11\%$	$3.6 \pm 0.10\%$	$3.5 \pm 0.10\%$
ρ	—	0.31 ± 0.01	0.58 ± 0.01	0.55 ± 0.01

Table 1: Estimated generalization performances and margins with confidence intervals for decision trees (C4.5), AdaBoost, Marginal AdaBoost and AdaBoost* on the toy data. All numbers are averaged over 200 splits into 100 training and 19900 test examples.

Adaboost and Noisy Data

- Examples with the largest gap between the label and the classifier prediction get the most weight
- What if the label is wrong?

Opitz: Synthetic data with 20% one-sided noisy labeling



Complexity of Weak Learner

- Complexity of strong classifier limited by complexities of weak learners
- Example:
  - Weak Learner: stubs → WL $Dim_{VC} = 2$
  - Input space: R^N → Strong $Dim_{VC} = N+1$
  - N+1 partitions can have arbitrary confidences assigned

Adaboost – Why does it work?

- Margins of the training examples

margin(x,y)= $\dfrac{y\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t}$.

- Positive only if correctly classified by H
- Confidence in prediction:

$$\hat{\Pr}\left[\text{margin}(x,y) \le \theta\right] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

- Qualitative Explanation of Effectiveness
  - Not Quantitative.

Bound on Generalization Error

Confidence of correct decision

VC dimension

$$\Pr_S[yf(x) \le \theta] + O\left(\frac{1}{\sqrt{m}}\left(\frac{d\log^2(m/d)}{\theta^2} + \log(1/\delta)\right)^{1/2}\right)$$

Confidence Margin

Number of training examples

Bound confidence term

To loose to be of practical value:

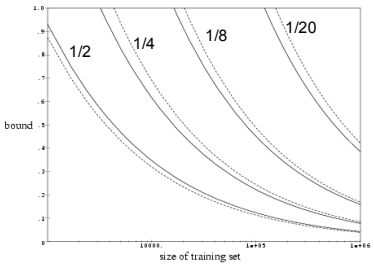
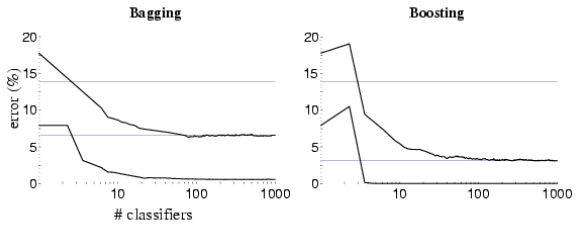


Figure 3: A few plots of the second and third terms in the bound given in Equation (8) (solid lines) and their approximation by the second term in Equation (9) (dotted lines). The horizontal axis denotes the number of training examples (with a logarithmic scale) and the vertical axis denotes the value of the bound. All plots are for $\delta = 0.01$ and $|\mathcal{H}| = 10^6$. Each pair of close lines corresponds to a different value of θ, counting the pairs from the upper right to the lower left, the values of θ are 1/20, 1/8, 1/4 and 1/2.

Immunity to Overfitting?

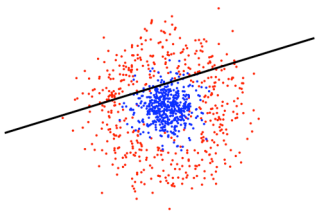


Algorithm recapitulation t = 1

Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$



Output the final classifier:

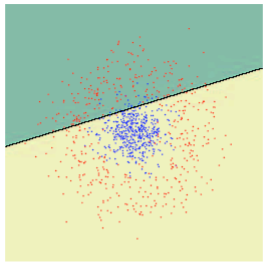
$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

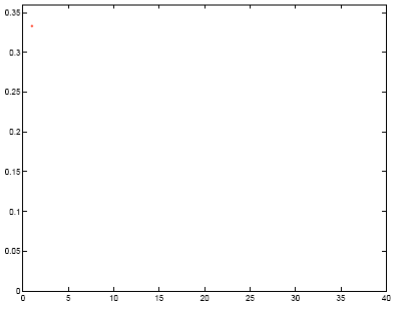
Algorithm recapitulation $t = 1$

Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$



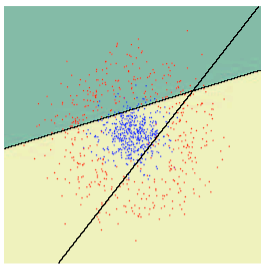


Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recap

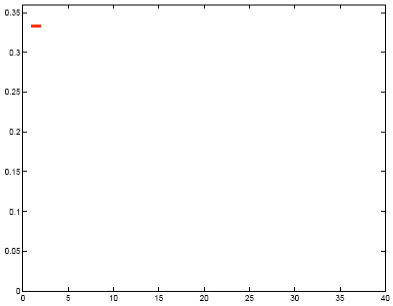
$t = 2$



Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

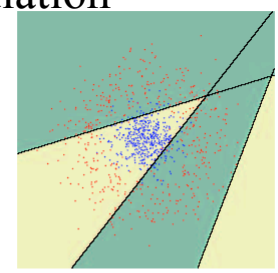


Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recapitulation

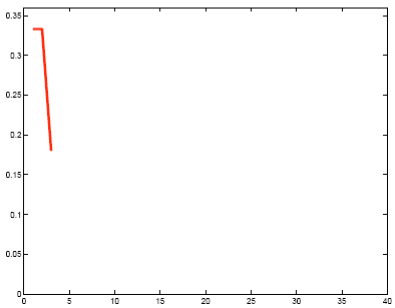
$t = 3$



Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

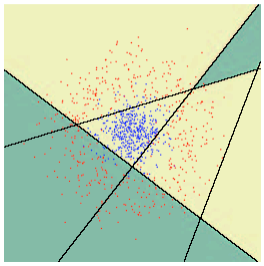


Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recapi

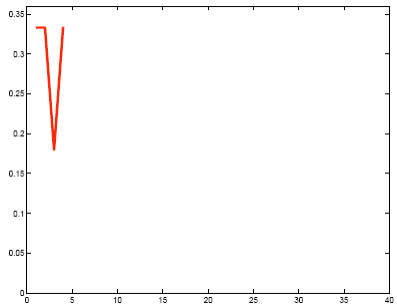
$t = 4$



Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

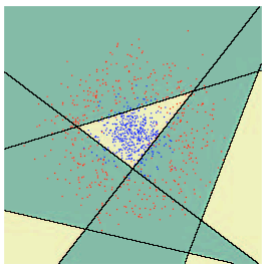


Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recap

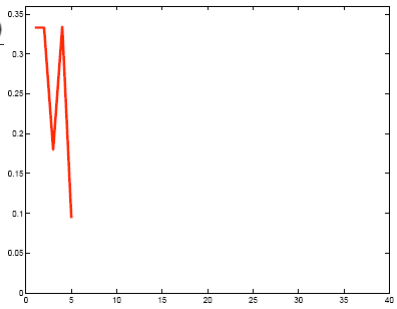
$t = 5$



Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$



Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recapi

Initialization...
For $t = 1, ..., T$:

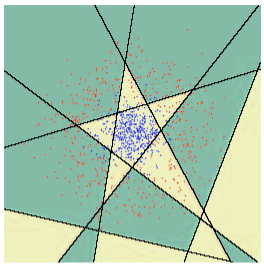
- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

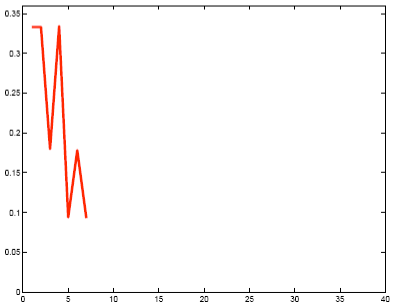
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

$t = 7$





Algorithm recap

Initialization...
For $t = 1, ..., T$:

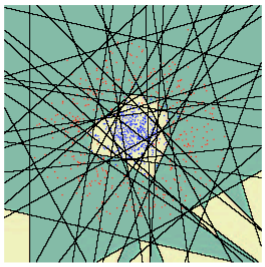
- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

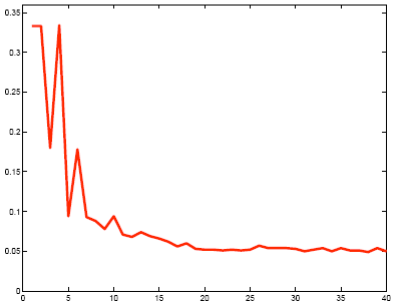
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

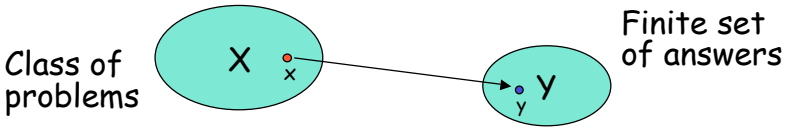
$t = 40$





Monte Carlo Algorithms and Ensemble Classification

Monte Carlo Stochastic Algorithms



A stochastic algorithm **MC** is Monte Carlo if:

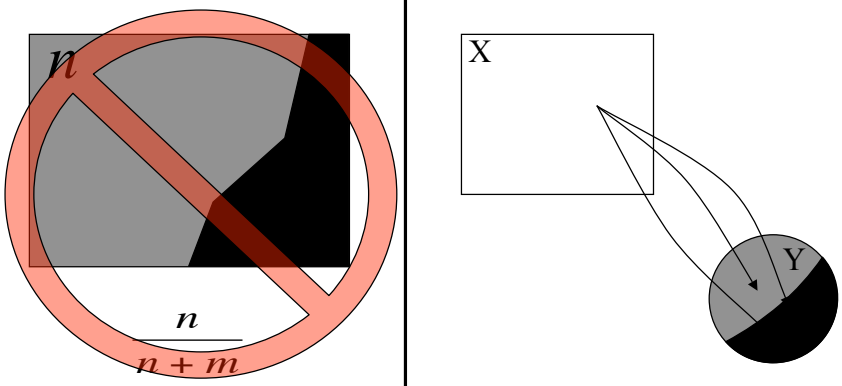
- Running MC on any $x \in X$ always yields an answer $y = MC(x)$, but this answer may (occasionally) be incorrect
- The probability of getting a correct answer is uniformly lower-bounded over the whole set X

MC is *p-correct* if for any $x \in X$: Pr {y is correct} $\geq p$

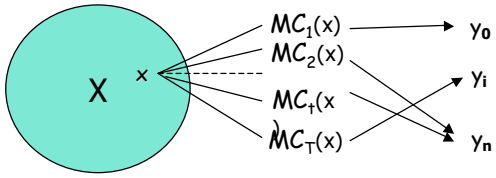
Error : $\varepsilon = 1 - p$ *Advantage*: $\gamma = p - 1/2 = 1/2 - \varepsilon$

P-Correctness

What does Pr{y is correct}= p mean?



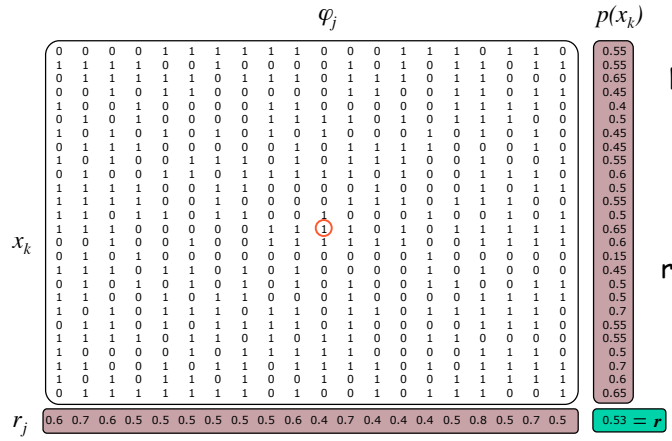
Advantage Amplification



Let x ∈ X be any problem. If we run MC repeatedly on x a number T of times, and we take the majority "vote" as answer, then MC's advantage can be increased as much as desired, provided that the following conditions are met :

1. Different calls to **MC** are independent
2. (**MC** is consistent)
3. **MC** is p-correct with p > 1/2

Basic Parameters



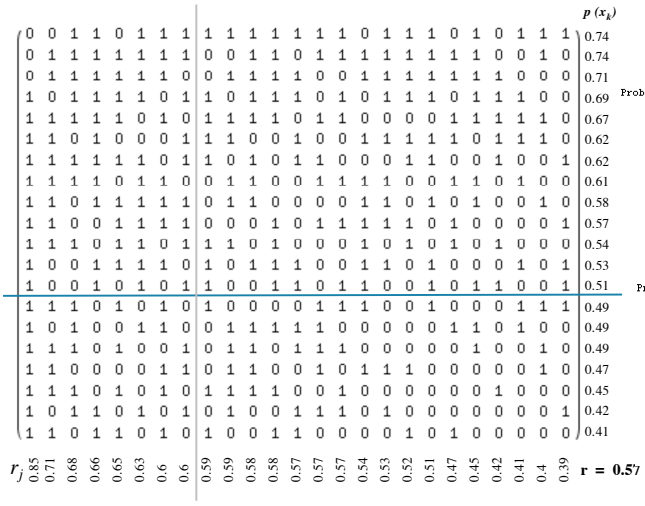
$$p(x_k) = \sum_{j=1}^{R} q_j \, p_j(x_k)$$

$$r_j = \sum_{k=1}^{N} d_k \, p_j(x_k)$$

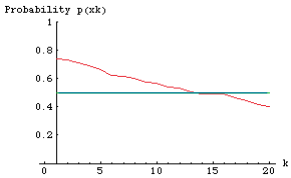
$$r = \sum_{j=1}^{R} \sum_{k=1}^{N} d_k \, q_j \, p_j(x_k)$$

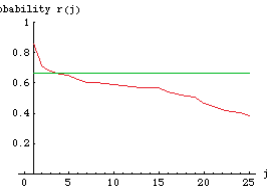
$m_{kj} = p_j(x_k) = \Pr\{\varphi_j(x_k) = w(x_k)\} \in \{0, 1\}$ --> Bayes error = 0

Matrix M (Sorted)



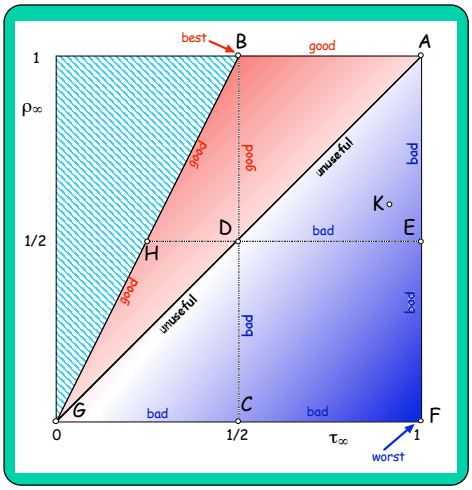
$||X_A|| = r_\infty = 0.66$



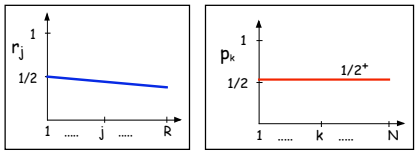


$t_\infty = \text{Max}_j \, r_j = 0.85$ $||F_B|| = c_\infty = 0.15$

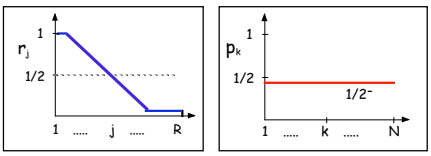
Perfect Information - Selection vs. Combination



Point B : Best



Point F : Worst



$2r - 1 \leq r_\infty < 2r$

Partial Information: Combining *T* hypotheses

Monte Carlo assumptions:

1. Different calls to **MC** are independent
2. **MC** is p-correct with p > 1/2

imply that:

1. Rows in M are treated independently
2. Multiple occurrences of x_k are classified independently

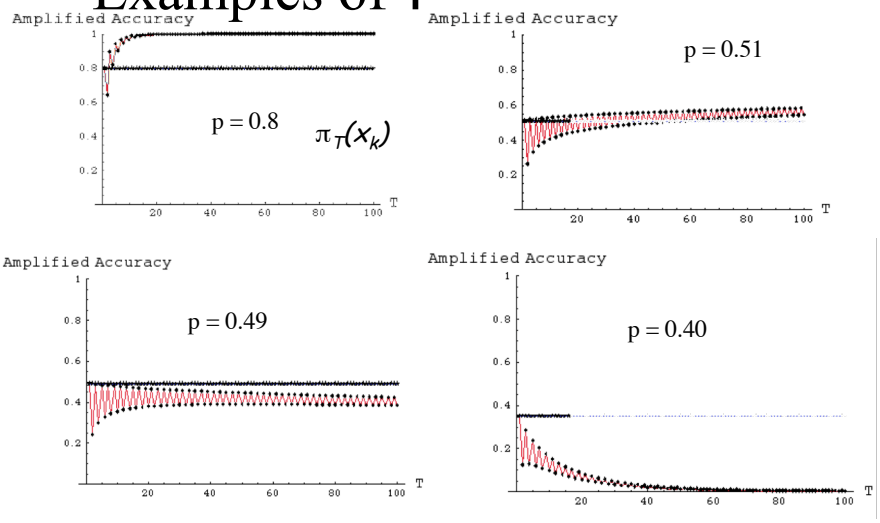
Probability of Correct Classification

$\pi_T(x_k)$ = Probability that x_k of x_k is correctly classified by a majority voting procedure over T extracted hypotheses

$$\pi_T(x_k) = \sum_{t=\left\lfloor \frac{T}{2}+1 \right\rfloor}^{T} \binom{T}{t} p(x_k)^t \left[1 - p(x_k)\right]^{T-t}$$

$$\begin{aligned} \mathrm{Lim}_{T \to \infty} \pi_T(x_k) &= 1 \quad \text{if } p > 1/2 \\ &= 0 \quad \text{if } p < 1/2 \\ &= 1/2 \quad \text{if } p = 1/2 \end{aligned}$$

Examples of π_T's behaviour

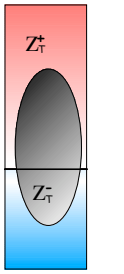


Accuracy r_T

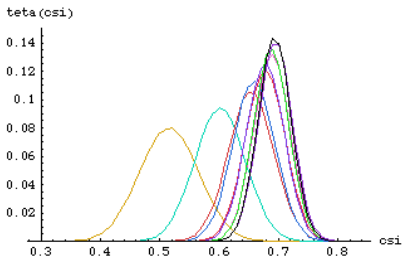
$$\rho_T = \rho_\infty - \rho_T^- + \rho_T^+$$

$$\lim_{T\to\infty} \mathbf{E}(\rho_T) = \rho_\infty$$

$$\lim_{T\to\infty} \mathbf{VAR}(\rho_T) = 0$$



Z_T
Y_A
$Y - Y_A$



$$\Theta_T(m) = \sum_{v=\mathrm{Max}\{0,\, m-M\rho_\infty\}}^{\min\{m,\,(1-\rho_\infty)M\}} \left[\sum_{Y_{M\rho_\infty - m+v} \subseteq Y_A} \left(\prod_{x_k \in Y_{M\rho_\infty - m+v}} [1-\pi_T(x_k)] \prod_{x_k \in Y_A - Y_{M\rho_\infty - m+v}} [\pi_T(x_k)] \right) \cdot \sum_{Y_v \subseteq Y - Y_A} \left(\prod_{x_k \in Y_v} \pi_T(x_k) \prod_{x_k \in Y - Y_A - Y_v} [1-\pi_T(x_k)] \right) \right]$$

$\Theta_T(m)$ can be approximated extremely well by a Gaussian

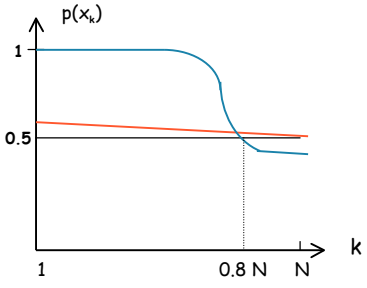
$\Theta_T(m)$ can be computed with a DP algorithm in $O(M^3)$ time

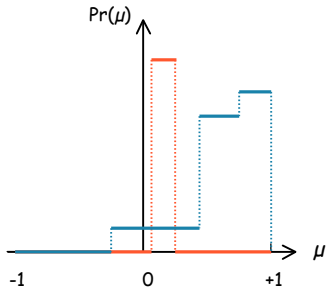
Margin

For binary classification, the following relation holds:

$$\mu(x_k) = 2p(x_k) - 1$$

$$\mu(x_k) > 0 \text{ iff } p(x_k) > 1/2$$





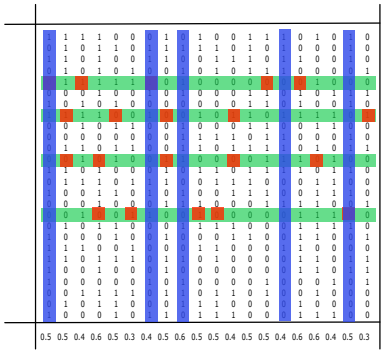
MC-Base EC works by Rows

Classify M (possibly duplicated) examples

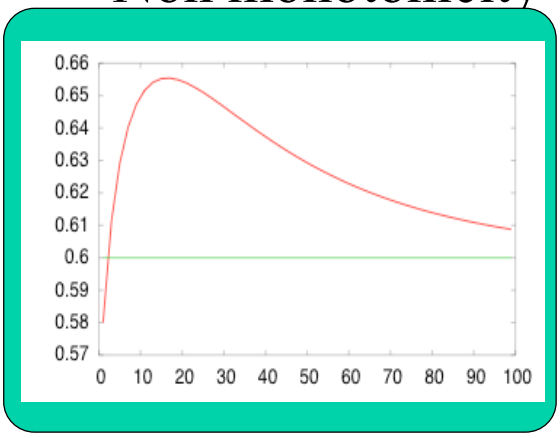
Monte Carlo and Bagging

- Selecting T columns forces **dependencies** among examples
- When T goes to **infinity**, Monte Carlo ad Bagging are equivalent
- When T is **finite**, Bagging is an approximation of a MC. The more "random" is the table, the closer are the two processes.

- The **variance** of Monte Carlo accuracy is provably lower than that of Bagging
- **Order correctness** on x_k corresponds to $p(x_k) > 1/2$

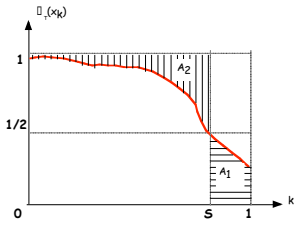


Non monotonicity of $E(\rho_T)$



For a given T:

Area A_1 > Area A_2



Monte Carlo and Bagging

- Selecting T columns forces **dependencies** among examples
- When T goes to **infinity**, Monte Carlo ad Bagging are equivalent
- When T is **finite**, Bagging is an approximation of a MC. The more "random" is the table, the closer are the two processes.
- Bagging's **Order correctness** on x_k corresponds to $p(x_k) > 1/2$

Bagging and MC-EC have the SAME mean accuracy
but DIFFERENT variances

Some Results

- Conditions for comparing model selection with model combination (when ensemble classification is likely to work)
- Upper and lower bound on asymptotic ensemble classification accuracy
- Link to "order correctness" and to the *margin,* and principled explanation of margin's role
- Exact and approximate probability distributions of ensemble classification accuracy (lower variance than bagging)
- Bias/variance decomposition of the error
- Suggestion for a notion of hypotheses "diversity"

Further Results

- Hypotheses in a composite classifier must have accuracy greater than 1/2 -> false
- High margins imply high ensemble classification accuracies -> false
- Increasing T, ensemble classification accuracy is monotonic (usually increases) -> false
- Ensemble classification behaviour CANNOT be predicted from the accuracies of the component hypotheses (except in trivial cases)

Biased Monte Carlo Algorithms

If MC(x) is y-biased, consistent and p-correct then the answer y is always correct

Let MC(x) = y

- If $x \notin A$, MC(x) is always correct, then y is indeed the correct answer
- If $x \in A$, MC(x) must necesarily be y, due to MC(x)'consistency

Let MC(x) = $z \neq y$

- If $x \notin A$, MC(x) is always correct, then z is correct
- If $x \in A$, MC(x) has made a mistake, because the correct answer is y. The probability of this event is not greater than (1-p), being MC(x) p-correct

Amplifying the Advantage of a Biased Algorithm

Let MC(x) run k times on the same instance x, and let $y_1, ..., y_k$ be the sequence of answers

- If $\exists i \mid y_i = y$, then y is the correct solution, even though it occurred just once
- If $\exists i,j \mid y_i \neq y_j$, the only possibility is that $x \in A$, because MC(x) is consistent; then, the correct solution is y
- If $\forall i \mid y_i = z \neq y$, the correct answer can still be y, and MC(x) has made k mistakes; but the probability of this event is at modt $(1-p)^k$

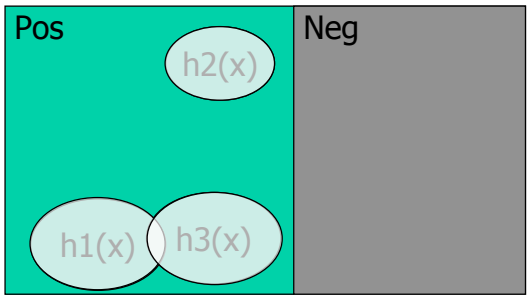
k Repetitions of a consistent, y-biased, p-correct Monte Carlo algorithm yield a consistent, y-biased, (1- (1-

Biased MC algorithms

MC	Biased MC
Majority Voting	At least one
$\varepsilon<.5$	$\varepsilon<1$
$T_{\Delta p}$	$T'_{\Delta p} << T_{\Delta p}$

$1-\varepsilon$	$1-\eta$	$T_{\Delta p}$ MC	$T_{\Delta p}$ BiasedMC
0.55	0.95	269	4
0.55	0.99	367	6
0.10	0.99	--	11

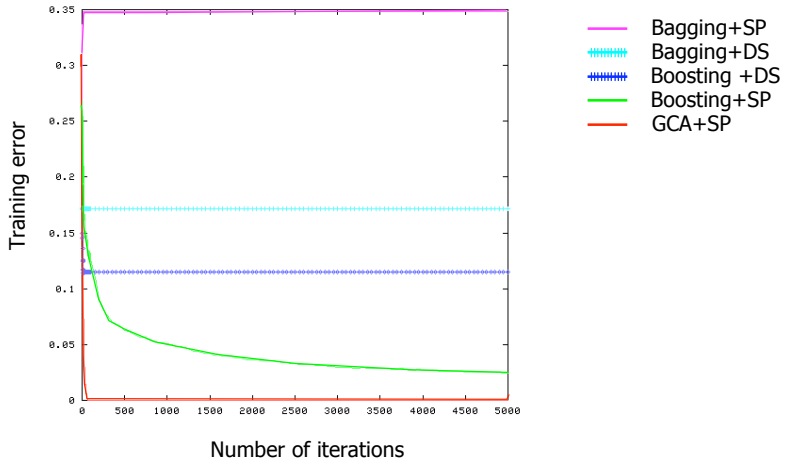
Biased MC and Ens. Learning



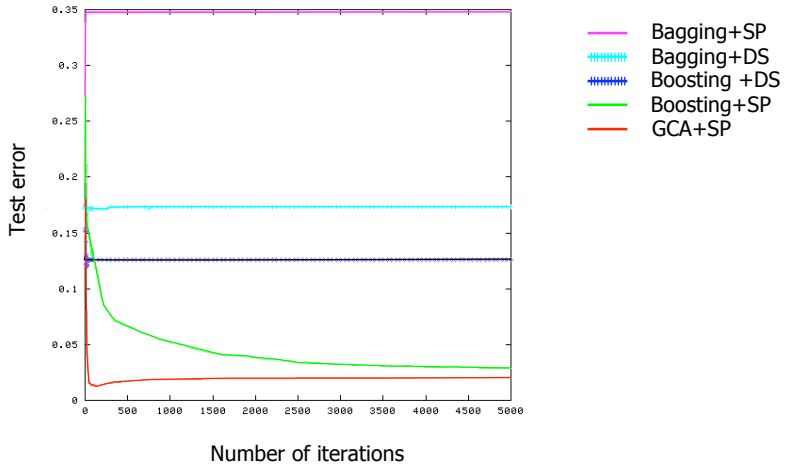
Greedy covering Strategy (AQ Like)

h_i make errors only on positive examples

Experiments



Experiments



Diversità

Diversity

- CLAIM → "Diverse" hypotheses imply high ensemble accuracy
- What is DIVERSITY ?
  - Many definitions of "diversity"
- INTUITION → Hypotheses must disagree in classifying examples

Analytical Study

$X = \{x_1, \ldots, x_k, \ldots, x_N\}$ $\quad$ $\Phi = \{\varphi_1, \ldots, \varphi_j, \ldots, \varphi_R\}$

	φ_1	...	φ_j	...	φ_R	
x_1	$p_1(x_1)$	...	$p_j(x_1)$	...	$p_R(x_1)$	$p(x_1)$
...	...					
x_k	$p_1(x_k)$	..	$p_j(x_k)$	...	$p_R(x_k)$	$p(x_k)$
...	...					
x_N	$p_1(x_N)$	...	$p_j(x_N)$	...	$p_R(x_N)$	$p(x_N)$
	r_1	...	r_j	...	r_R	r

$$p(x_k) = \frac{1}{R}\sum_{j=1}^{R} p_j(x_k)$$

$$r_j = \frac{1}{N}\sum_{k=1}^{N} p_j(x_k)$$

$$r = \frac{1}{NR}\sum_{k=1}^{N}\sum_{j=1}^{R} p_j(x_k)$$

Local Measures of Diversity

Diversity between two hypotheses → Average over all pairs

- Correlation coefficient C
- Youle's Q statistics → $Q = \mathbf{E}[Q_{ij}] = \mathbf{E}\left[\frac{N_{11}(i,j)N_{00}(i,j) - N_{10}(i,j)N_{01}(i,j)}{N_{11}(i,j)N_{00}(i,j) + N_{10}(i,j)N_{01}(i,j)}\right]$
- Disagreement Dis → $Dis = \mathbf{E}[Dis_{ij}] = \frac{1}{N}\mathbf{E}\left[N_{10}(i,j) + N_{01}(i,j)\right]$
- Double-Fault DF → $DF = \mathbf{E}[DF_{ij}] = \frac{1}{N}\mathbf{E}\left[N_{00}(i,j)\right]$

Global Measures of Diversity

- Entropy E
- Kohavi-Wolpert Variance KW
- Interrater Agreement
- Difficulty
- Generalized Diversity GD
- Coincident Failure Diversity CFD

Relations among Parameters

$$\kappa = \frac{R\theta - 1}{R - 1}$$

$$c = 1 - \frac{R\gamma}{(R-1)r(1-r)} = \kappa$$

$$GD = \frac{R}{R-1}(1-\theta)$$

$$\tau = 1 - GD$$

$$CFD - DFD = \frac{\gamma}{R-1}$$

KW = γ

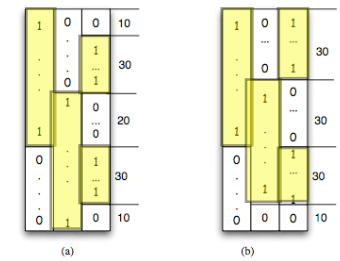
Main Finding

NO MEASURE IS RELATED TO
ENSEMBLE CLASSIFIER'S ACCURACY
IN ANY USEFUL WAY

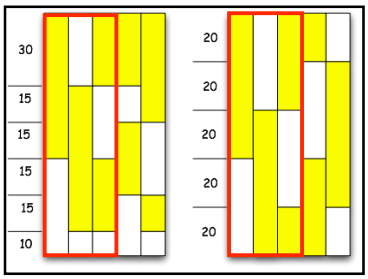
Further Reasons for Pessimism

Hypotheses must be "diverse" but "not too much"

The best overlapping for T may not be the best for (T+k)



$\rho_3 = 0.80$ $\rho_3 = 0.90$



$\rho_3 = 0.90$
$\rho_5 = 0.90$

$\rho_3 = 0.80$
$\rho_5 = 1.00$

Ambiguity

$$A_j(x_k) = \begin{cases} 1 & \text{if } \varphi_j(x_k) \neq H_R(x_k) \\ 0 & \text{if } \varphi_j(x_k) = H_R(x_k) \end{cases}$$

$$\rho_R = A + r - \frac{2}{N} \sum_{k=1+S}^{N} p(x_k)$$